Exact rounding for geometric constructions

Hervé Brönnimann, Sylvain Pion

Submitted to Scientific Computing, Computer Arithmetic, and Validated Numerics (SCAN) 1997.

Exact rounding for geometric constructions

Hervé Brönnimann, Sylvain Pion*

INRIA Sophia-Antipolis,
2004, Route des Lucioles, B.P. 93,
06902 Sophia-Antipolis Cedex, France.

Abstract: Exact rounding is provided for elementary floating-point arithmetic operations (e.g. in the IEEE standard). Many authors have felt that it should be provided for other operations, in particular for geometric constructions. We show how one may round modular representation of numbers to the closest f.p. representable number, and demonstrate how it can be applied to a variety of geometric constructions. Our methods use only single precision; they produce compact, efficient, and highly parallelizable code. We suggest that they can be applied in other settings when exact computations interact closely with rounded representations.

Keywords: computational geometry, exact arithmetic, robustness, modular computations, single precision, Residue Number Systems (RNS)

1 Introduction

The success of floating point (f.p.) arithmetic may largely be attributed to the exact rounding featured (specified e.g. in the IEEE 754 double precision standard [7]). This enables to obtain error estimates for elementary operations, and by roundoff error analysis, for more complex operations. Geometric constructions, however, do not seem to possess the kind of numerical stability that makes numerical computations so prone to analyze (although some attemps have been made, as in [4]). Therefore, geometric algorithms easily suffer from robustness problems due to the limited precision of the computations [8]. Moreover, the input to an algorithm is often a geometric structure output by another algorithm, for which we cannot keep an exact representation. Rounding this structure using the standard f.p. approximation creates inconsistencies between the combinatorial and geometric representation. In relatively few steps of this cascading process, the geometric algorithm is almost sure to fail because of accuracy problems [12]. Exact rounding of geometric structures (both geometrically and combinatorially) is thus a natural entension of exact rounding with f.p. arithmetic for devising and analyzing robust implementations of geometric algorithms.

We continue the investigation of modular arithmetic for exact geometric predicates started in [2]. In [2], techniques similar to those of this paper are developed for the problem of determining the sign of an integer given its residues. The techniques are then used for devising exact geometric predicates, which can then ensure robustness of geometric algorithms. Here, we are concerned with rounding the geometric objects to a grid (as in [12] and references therein). Our techniques can handle grids with integer, floating-point, or even polar coordinates. We are not concerned with the combinatorial part of the rounding, a difficult problem in its own right (see [12] and references therein). Rather, we focus on the numerical part of the problem.

Exact rounding can can be achieved via full-precision computations, but this can be expensive. It can also be achieved by the LN approach [6] for integers, or the techniques of Schewchuk [13] for floating point numbers. However, the size of the code produced by these techniques grows quadratically with the complexity of the construction. This can be prohibitive already for vertices of Voronoi diagrams of line segments. Using our techniques, we may spend quadratic time in the worst case, but the code remains compact and is easily parallelizable (with a speedup almost linear in the number of processors).

2 Closest f.p. approximation

Floating point (f.p.) computations. Our model of a computer is that of a f.p. processor that performs operations at unit cost by using b-bit precision (e.g., in the IEEE 754 double precision standard, we have $b = 53$). It is a realistic model as it covers the case of most workstations used in research and industry [7, 10, 13].

To be able to discuss the properties of f.p. arithmetic, it is convenient to introduce the following notation [13]:

*This research was partially supported by the ESPRIT IV LTR Project No. 21957 (CGAL).

given any real number x, it is *representable*[1] over b bits if $x = 0$ or if $x2^{-\lfloor \log x \rfloor + b}$ is an integer;[2] $\widetilde{x}$ denotes the representable f.p. number closest to x (with any tie-breaking rule if x is right in-between two representable numbers), and $\mathrm{ulp}(x)$ denotes the *unit in the last place*, that is, $2^{\lfloor \log |x| - b \rfloor}$ if $x \neq 0$, and 0 otherwise. We will use mainly one basic property of f.p. arithmetic on such a computer: for all four arithmetic operations, the absolute error in computing an operation that returns x is $\frac{1}{2}\mathrm{ulp}(x)$. In particular, operations performed on pairs of integers smaller than 2^b are performed exactly as long as the result is also smaller than 2^b.

Modular computations. Let $m_1, \ldots, m_k$ be k pairwise relatively prime integers and let $m = \prod_i m_i$. For any number x (not necessarily an integer), we let $x_i = x \bmod m_i$ be the only number in the range $\left[-\frac{m_i}{2}, \frac{m_i}{2}\right)$ such that $x_i - x$ is a multiple of m_i. (This operation is always among the standard operations because it is needed for reducing the arguments of periodic functions.) As shown in [2], arithmetic modulo m_i can be performed on integers by using f.p. arithmetic with b-bit precision, provided that $m_i \leq 2^{b/2+1}$.

This operation can be extended modulo an f.p. numbers as follows: an f.p. number x is truncated to a non-null f.p. number y and the result is defined as $x - \lceil x/y \rfloor y$. Therefore, $x \bmod m_i$ is the result of truncating x to m_i, and the (signed) fractional part $\mathrm{frac}(x)$ of x is the result of truncating x to 1. Note that the result of truncating x to a power of two is always representable if x is representable.

The Chinese remainder theorem,[1, 10] shows that integers in the range $\left[-\frac{m}{2}, \frac{m}{2}\right)$ can be uniquely represented by their residues $x_i = x \bmod m_i$; we speak of a k-modular representation. Rationals can then be represented by modular representations of their numerator and denominator, and algebraic numbers by a polynomial with integral coefficients and a rational isolating interval. With this notation, we can express the basic problem considered in this paper.

Problem 1 *Let x be a number represented as above. Compute a f.p. number $\widetilde{x}$ such that $|\widetilde{x} - x| \leq \frac{1}{2}ulp(x)$ by using only f.p. operations with b-bit precision.*

2.1 The case of integers

When x is an integer in the range $\left[-\frac{m}{2}, \frac{m}{2}\right)$ represented by its residues $x_i = x \bmod m_i$, $i = 1, \ldots, k$, the value of x can be retrieved by the following formula, due to Lagrange:

$$x = \left(\sum_{i=1}^{k} ((x_i w_i) \bmod m_i)\, v_i \right) \bmod m. \qquad (1)$$

where $v_i = m/m_i = \prod_{j \neq i} m_j$, and $w_i = v_i^{-1} \bmod m_i$. To treat the case of $j < k$ moduli, we introduce the notation:

$$\begin{aligned} m^{(j)} &= \prod_{1 \leq i \leq j} m_i, \\ v_i^{(j)} &= \prod_{\substack{1 \leq \ell \leq j \\ \ell \neq i}} m_\ell, \\ w_i^{(j)} &= \left(v_i^{(j)}\right)^{-1} \bmod m_i, \end{aligned}$$

We will therefore compute with fixed b-bit precision the following sums for $j = k, \ldots, 1$:

$$R^{(j)} = \left(\sum_{i=1}^{j} (x_i w_i^{(j)} \bmod m_i) v_i^{(j)} \right) \bmod m^{(j)}.$$

With the properties of f.p. arithmetic , it is not hard to prove that the computed value $R^{(k)}$ approximates x with an absolute error $E^{(k)} = x - R^{(k)}$ bounded by $\varepsilon_k m^{(k)}$ where $\varepsilon_k = 3k2^{-b-1}$ [2]. The key idea is that we can compute an exact $(k-1)$-representation of this error since $E^{(k)} \bmod m_j = (x_j - R^{(k)}) \bmod m_j$. We can again compute an approximation $R^{(k-1)}$ of $E^{(k)}$, by applying Lagrange's method with these $k-1$ residues and computing $R^{(k-1)}$ with an absolute error $E^{(k-1)}$ bounded by $(\varepsilon_{k-1} + 2^{-b-1})m^{(k-1)}$. Recursively, for a decreasing $j = k-1, \ldots, 1$, we can compute a $(j-1)$-representation of $E^{(j)}$ by $E^{(j)} \bmod m_i = (R^{(j+1)} - R^{(j)}) \bmod m_i$ for each $i = 1, \ldots, j$. An approximation $R^{(j-1)}$ of $E^{(j)}$ is obtained by Lagrange's method on $j-1$ moduli with absolute error $E^{(j-1)}$ bounded in magnitude by $(\varepsilon_{j-1} + 2^{-b-1})m^{(j-1)}$. For any $j = k-1, \ldots, 0$, we have

$$x = R^{(k)} + \cdots + R^{(j)} + E^{(j)}.$$

Since the non-zero $R^{(i)}$'s are decreasing by a factor $\varepsilon_k < 1/2$, a good way to compute their sum is to perform the additions in the order $R^{(k)} + (R^{(k-1)} + (\cdots + (R^{(j+1)} + R^{(j)}) \cdots))$. With b-bit f.p. precision, the result Z_j approximates $\sum_{i=j}^{k} R^{(i)}$ with an absolute error bounded by $\mathrm{ulp}(Z_j)$. Moreover, truncating the leading bits of $R^{(i)}$ to $\mathrm{ulp}(Z_j)$ yields a sum z_j truncated to $\mathrm{ulp}(Z_j)$ such that $Z_j + z_j$ approximates $\sum_{i=j}^{k} R^{(i)}$ with an absolute error bounded by $\mathrm{ulp}(z_j)$. (Note that truncating $R^{(i)}$ to $\mathrm{ulp}(Z_j)$ can be achieved by computing $(R^{(i)} + Z_j) - Z_j$ with b-bit f.p. precision.) Then $Z_j + z_j$ approximates x with an absolute error bounded by

$$\mathrm{ulp}(z_j) + |E^{(j)}| \leq \mathrm{ulp}(z_j) + (\varepsilon_j + 2^{-b-1})m^{(j)}.$$

[1] We systematically ignore underflows and overflows, by assuming that the range of exponent is large enough. A few modern packages now provide f.p. arithmetic with the exponent stored in a separate integers, which extends the IEEE 754 double precision standard by quite a lot.

[2] All logarithms in this paper are base 2.

If $Z_j + z_j$ truncated to $\frac{1}{2}\text{ulp}(Z_j + z_j)$ is distant from $\frac{1}{4}\text{ulp}(Z_j + z_j)$ by at least $\text{ulp}(z_j) + (\varepsilon_j + 2^{-b-1})m^{(j)}$, then $Z_j + z_j$ can be rounded to the closest f.p. representable number and yields the desired $\widetilde{x}$.

A minor difficulty arises because it is hard to compute $R^{(j)} \bmod m_i$: in the worst case, this may take $\lfloor \frac{1}{b} \log R^{(k)} \rfloor$ f.p. divisions. However, this is one single instruction in the IEEE 754 standard and we will account for it as a single instruction. (Note that although $R^{(k)}$ is a multiple of $m(k)$ by $S^{(k)}$, $S^{(k)}$ is not an integer, and there is roundoff error in computing the product $S^{(k)}m^{(k)}$, so we cannot take advantage of this to compute $R^{(j)} \bmod m_i$.)

This leads to the following algorithm.

Algorithm 1 : Compute $\widetilde{x}$ knowing $x_i = x \bmod m_i$

Precomputed data: m_j, $\widetilde{m^{(j)}}$, $w_i^{(j)}$, η_j
Input: k *and* $x_i \in \left[-\frac{m_i}{2}, \frac{m_i}{2}\right)$, *for all* $1 \le i \le k$
Output: *sign of* x, *the unique solution of* $x_i = x \bmod m_i$ *in* $\left[-\frac{m^{(k)}}{2}, \frac{m^{(k)}}{2}\right)$
Precondition: $|x| + \eta_k \le \frac{m^{(k)}}{2}$

1. Let $j \leftarrow k$,
$R^{(k)} \leftarrow \left(\sum_{i=1}^{k} (x_i w_i^{(k)} \bmod m_i) v_i^{(k)}\right) \bmod m^{(k)}$,
2. Repeat $j \leftarrow j - 1$,
$x_i \leftarrow (x_i - R^{(j+1)}) \bmod m_i$ *for all* $i = 1, \ldots, j$
$R^{(j)} \leftarrow frac\left(\sum_{i=1}^{j} \frac{x_i w_i^{(j)} \bmod m_i}{m_i}\right) m^{(j)}$
$Z_j \leftarrow R^{(k)} + (R^{(k-1)} + (\cdots + (R^{(j+2)} + R^{(j+1)}) \cdots))$
$z_j \leftarrow R^{(k)} + (R^{(k-1)} + (\cdots + (R^{(j+2)} + R^{(j+1)}) \cdots))$
truncated to $ulp(Z_j)$
$x' \leftarrow (Z_j + z_j)$ *truncated to* $\frac{1}{2} ulp(Z_j + z_j)$
until $j = 0$ *or* $\left|x' - \frac{1}{4} ulp(Z_j + z_j)\right| > ulp(z_j) + (\varepsilon_j + 2^{-b-1})m^{(j)}$
3. return $Z_j + z_j$

Although we do not give a complete analysis of the algorithm in this abstract, it is easy to see that, given k residues, the algorithm will perform $O(k^2)$ f.p. operations in the worst case, and only $O(k)$ operations in most practical cases.

2.2 The case of rationals

We can obtain the closest f.p. approximation of a quotient $z = x/y$ where x and y are given by their k-modular representations. Indeed, when computing z' by performing $\widetilde{x}/\widetilde{y}$ with b-bit precision, only the last (at most) three bits are incorrect. The correct bits can be determined by at most three binary searches: if z' is expressed exactly as u/v with integral u and v (for instance, we could take $v = 1/\text{ulp}(z)$ if z' is not an integer), then we can compute modular representations of u and v and compare z' with z by computing the sign of $vx - uy$ with the help of the algorithm of [2].

2.3 The case of algebraic numbers

Algebraic numbers can be manipulated explicitly with the Sturm or elimination theories, which requires a lot of sign evaluations and is dealt with easily with modular computations [2, 9]. A closest f.p. representation can be found by shrinking the isolating interval using binary search at binary rationals. Due to lack of space, we omit the discussion of this section. (See also [9] for applications in solid modeling.)

3 Geometric constructions

Common geometric constructions can be done with rationals and algebraic operations. Rounding these constructions to the grid can then be done using the techniques of the previous section. We discuss three such constructions to illustrate our approach. Although some of these examples can be handled by other techniques [5, 6, 13], the techniques proposed here are more effective with more complex constructions.

In order to avoid the representation of rationals, we use the homogeneous representation (a_x, a_y, a_w) of a point a in the plane. Exact rounding therefore computes the rounded homogeneous coordinates from the modular homogeneous coordinates. Cartesian representation with exact rounding can also be obtained from the modular homogeneous coordinates (see section 2.2). Given points a, b, c, their determinant is denoted by

$$[a, b, c] = \begin{vmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{vmatrix}.$$

In [2], it is shown how to compute a modular representation of such determinants very efficiently.

Exact rounding of the following constructions to the Cartesian or homogeneous grid can be obtained by using the techniques of the previous section.

3.1 Polygon placement

Computing Minkowski sums is a basic operation in polygon placement []. The Minkowski sum of two polygons A and B is defined as

$$A \bigoplus B = \{a + b, \ \ a \in A, \ b \in B\}.$$

If the polygons are convex, their sum is convex and the vertices are obtained by adding some vertices of A and B. Adding two points in homogeneous coordinates is done by

$$a \oplus b = (a_x b_w + a_w b_x, a_y b_w + a_w b_y, a_w b_w).$$

Thus the vertices of the sum can be rounded exactly.

3.2 Line intersections

Computing the intersection of two lines is a central problem in problems that output arrangements of lines and line segments []. Given four points, a, b, c, d, the intersection of the lines ab and cd is given in homogeneous coordinates by $[b, c, d]a - [a, c, d]b$ where multiplication and substraction are the standard operations on vectors.

3.3 Voronoi diagrams

Let us introduce the lifting φ that gives any point an additional coordinate $z = x^2 + y^2$; in homogeneous representation, $\varphi(a) = (a_x a_w, a_y a_w, a_x^2 + a_y^2, a_w^2)$.

Given three points a, b, c, let $d = (d_x, d_y, d_z, d_w)$ be the wedge product $a \wedge b \wedge c$, that is, the minors of the following 4×3 matrix:

$$\begin{pmatrix} a_x a_w & a_y a_w & a_z & a_w^2 \\ b_x b_w & b_y b_w & b_z & b_w^2 \\ c_x c_w & c_y c_w & c_z & a_w^2 \end{pmatrix}.$$

The center of their circumscribed circle is given in homogeneous coordinates by $(d_x, d_y, -2d_z)$, and the radius is given by

$$\frac{d_x^2 + d_y^2 - 4d_w}{4d_z^2}.$$

This simple example shows that the complexity of the computations can quickly become a problem. For Voronoi diagram of a convex polygon, the vertices require even more complex computations. For Voronoi of line segments, the vertices do not have rational coordinates: we must have recourse to algebraic computations.

3.4 Polygon and solid modeling

In [9, 11], it is shown how robust solid modeling can be achieved, by computing with approximations [11], or by using modular arithmetic for symbolic algebraic predicates [9]. Their constructions can be approximated using our techniques, improving the approximations. It is not clear however how this is to be incorporated in their algorithms.

4 Conclusion

The main properties of exact rounding are canonicity (a number has a uniquely defined approximation) and monotonicity (approximations have the same order, with strict inequalities becoming large inequalities). Exact rounding is provided by f.p. arithmetic on modern computers for the standard arithmetic operations. The techniques of this paper extend exact rounding for more complex computations (determinants, algebraic manipulations). We apply them to geometric constructions. They are more efficient compared to available methods for complex constructions, such as the last two given in section 3. It is conceivable that they find applications in other fields where hybrid arithmetic demand exact representation of numbers together with a provably accurate floating-point approximation.

As far as geometric applications are concerned, a major open problem is to generalize the combinatorial part of geometric rounding to higher dimensions (solid 3D modeling, curved surfaces, etc.).

References

[1] A. V. Aho and J. E. Hopcroft and J. D. Ullman. *The Design And Analysis Of Computer Algorithms*. Addison-Wesley, Reading, MA, 1974.

[2] H. Brönnimann, I.Z. Emiris, V.E. Pan, S. Pion. Efficient exact evaluation of signs of determinants. To appear in *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, Nice, 1997.

[3] B. Buchberger, G.E. Collins, and R. Loos, editors. *Computer Algebra: Symbolic and Algebraic Computation*. Springer, Wien, 2nd edition, 1982.

[4] S. Fortune. Numerical stability of algorithms for 2-d Delaunay triangulations and Voronoi diagrams. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 83–92, 1992.

[5] S. Fortune and C. J. Van Wyk. Efficient exact arithmetic for computational geometry. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 163–172, 1993.

[6] S. Fortune, C.J. Van Wyk. Static analysis yields efficient exact integer arithmetic for computational geometry. *ACM Trans. on Graphics*, 1996.

[7] D. Goldberg. What every computer scientist should know about floating point arithmetic. *ACM Comput. Surv.* 32(1):5–48, 1991.

[8] C. M. Hoffmann. The problem of accuracy and robustness in geometric computation. Report CSD-TR-771, Dept. Comput. Sci., Purdue Univ., West Lafayette, IN, 1988.

[9] J. Keyser, S. Krishnan, and D. Manocha. Efficient B-rep generation of low degree sculptured solids using exact arithmetic. Technical Report 40, Dept. Computer Science, Univ. N. Carolina, Chapel Hill, 1996.

[10] D.E. Knuth. *The Art of Computer Programming: Seminumerical Algorithms*, volume 2. Addison-Wesley, Reading, Massachusetts, 1981 and 1997.

[11] V. Milenkovic. Robust polygon modeling. *Comput. Aided Design* 25(9):546–566, 1993.

[12] V. Milenkovic. Shortest Path Geometric Rounding. newblock Submitted to *Algorithmica, Special issue on implementations of geometric algorithms, 1997.*

[13] J.R. Shewchuk. Robust adaptive floating-point geometric predicates. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages 141–150, 1996.

[14] C. Yap. Towards exact geometric computation. *Comput. Geom. Theory Appl.*, 7:3–23, 1997.

[15] C. K. Yap. Exact computational geometry and tolerancing metrology. In D. Avis and J. Bose, editors, *Snapshots of Computational and Discrete Geometry, Vol. 3, Tech. Rep. SOCS-94.50*. McGill School of Comp. Sci., 1995.

[16] C. K. Yap and T. Dubhe. The exact computation paradigm. In D. Du and F. Hwang, editors, *Computing in Euclidean Geometry*. World Scientific Press, 1995.